Wipermania: An All You Can Wipe Buffet

Stories

The latest cybersecurity trends, best practices, security vulnerabilities, and more

By Max Kersten · November 15, 2022

In early 2022, Ukrainian companies were struck by multiple destructive wipers, attacking various organizations across sectors. This raised questions about the usage and impact of "digital weapons" within the security community, even though wipers themselves weren't new. The infamous Shamoon wiper dates back more than a decade ago. How can it be that wipers were as effective a decade ago as they are in the present day? What has changed? And what remains the same?

This blog will include an analysis of more than twenty recent wiper families, their trends, techniques, and overlap with other wipers. Reused code and techniques may link several wipers to the same actor, although the mere presence of such a link often leads to a hasty conclusion. This is not a generic, run-of-the-mill comparison of malware families, as it includes technical aspects of the analyzed wipers, along with a focus on both the high- and low-level aspects of the destructive software.

This analysis does not only focus on the wipers used against Ukrainian victims, but also more generic wipers that were found in the wild around the same time. The parallels and differences between the targeted and generic wipers provide several interesting insights.

To conclude, defensive tips with regards to wipers will be given, allowing anyone to reduce the impact of future wiper related incidents.

Before diving into the topic, I'd like to thank all researchers and their organizations for their openness and willingness to share these reports publicly, allowing both the general public and other researchers to work with their analysis, rather than duplicating efforts.

Fear, uncertainty, and doubt

Before moving on to the wipers, a brief word about the fear, uncertainty, and doubt (often abbreviated as FUD) which is used in some reports. Earlier this year, we briefly touched on this subject in our War, Weapons, and Wipers blog. Elaborating on this, the expressed urgency and impact of digital attacks and the subsequently used malware varies per source, but hot takes are often made during times of crises. If the majority of malware campaigns are met with such fearmongering, the eventual effect will be detrimental. In short, to cry wolf: claiming all campaigns are equally impactful for everybody can limit the response from organizations, thus limiting the effect of the warnings overall.

This is not to say that urgent calls regarding malware campaigns are always spreading fear, uncertainty, and doubt. It is meant to say that context and nuance are crucial, especially when tensions are high.

Defining wipers

It is important to clearly specify what a wiper is, as the definition isn't set in stone. The more traditional definition refers to malware with the intent to cripple the victim's system. Malware with multiple features can also be used to destroy a system using a wiper component.

There is, however, also an unintended wiper use case in some ransomware cases. If the ransomware's encryption is faulty, there is no recovery option, or if the threat actor who deployed the ransomware cannot be reached, the ransomed system remains dysfunctional. In some cases, the email addresses of actors are blocked, or their website is taken offline, which might make it impossible to obtain a decryption key.

Third, a more obscure wiper variant: fake ransomware. Using ransomware as a cover, the malware might pretend the system is ransomed, whereas there never was the intention to decrypt the data in the first place. Properly encrypting a file system, as also noted in the paragraph above, is equal to wiping the data of said system, as it remains inaccessible in both cases.

In this blog, malware which has the intention to wipe a device is considered a wiper, be it as a part of malware or as the malware's sole purpose.

Wipers old and new

A decade ago, the Shamoon wiper wreaked havoc within Saudi Aramco's network as it wiped numerous machines. Over the past 10 years, security measures have greatly improved, as have the adversarial tactics and techniques. The constant cat and mouse game between blue teams and their adversaries now iterates faster than before. What once started with the physical mail of a new floppy to update a system can now be performed in near real-time. And yet, old techniques prove to be valuable, even if only for a short period of time. Afterall, attackers don't need much time to capitalize.

Achieving goals

The most important aspect of a wiper attack is to understand the goal and motive of the attack. While ransomware operators are generally financially motivated, wipers are used differently. There is no direct monetary benefit from wiping machines at corporations or governments. The benefits are often indirect, such as a competitive advantage as a competitor has been shut down, be it temporarily or permanently.

The term "competitor" in the paragraph above should be taken in a broad sense. It can mean someone who has a competing business, which the actor tries to distort with the help of illegal means. Another meaning can be a competitive business or government branch. Ukraine is a prime example of the latter case, and one of the first cases where multiple different wipers were used preceding and during a military invasion with the goal to help the attacker's military.

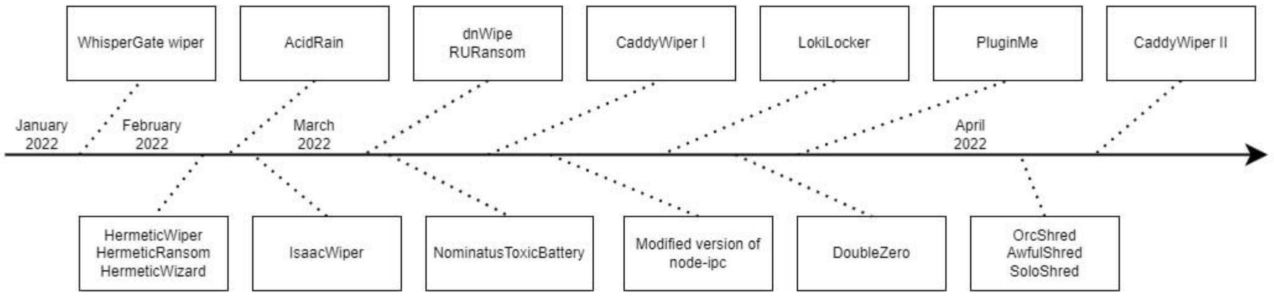
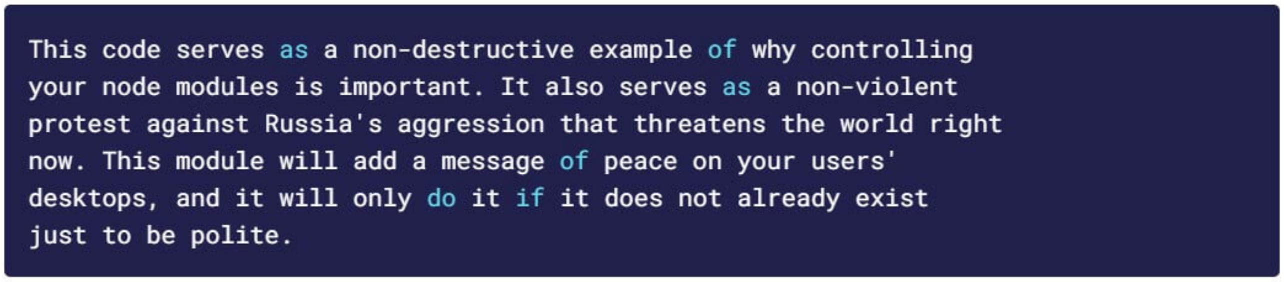


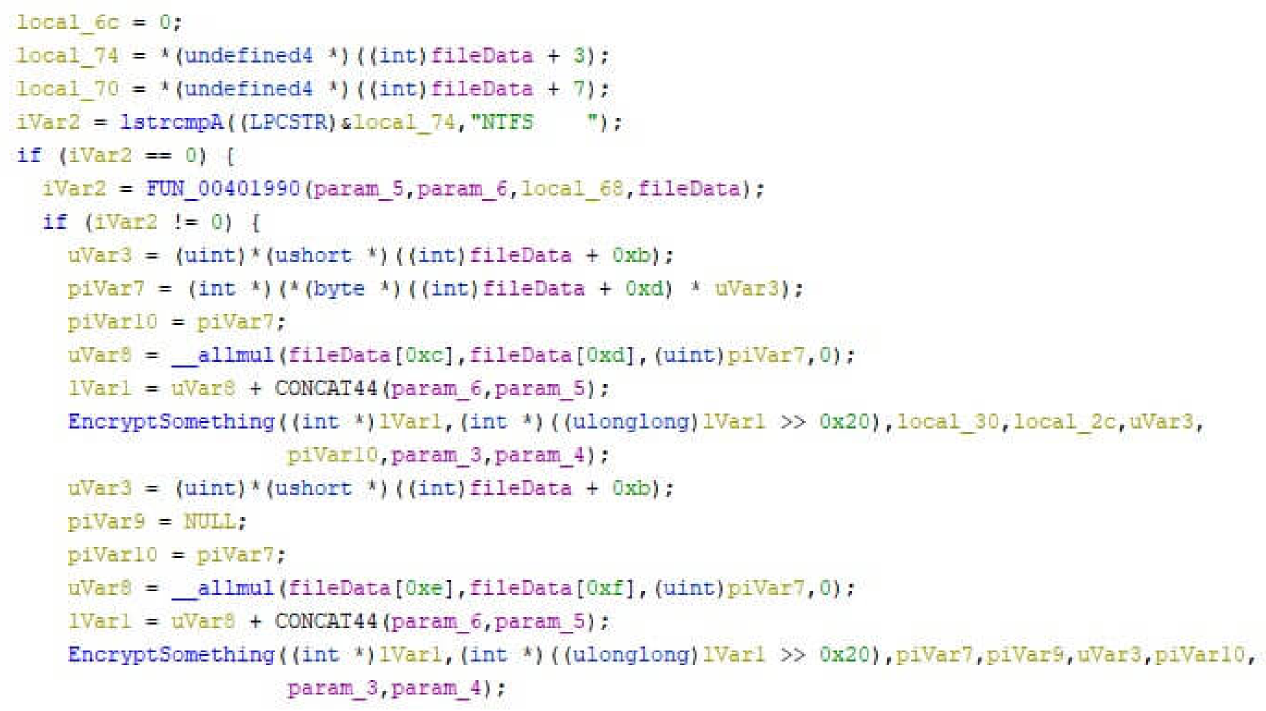
Figure 1 - A timeline of wiper occurrences, ranging from January through April in 2022
While some of the wipers in the timeline above were attributed to Russia, some of the wipers were made by activists, be it opponents or proponents of the war. One such example is a NodeJS inter process communication package known as node-ipc, as observed by Snyk. This package is commonly used, where the malicious version used to overwrite all files in the current project with the heart emoji if the user's IP indicates their geolocation refers to Russia. Additional changes, such as the inclusion of the peacenotwar package, created by the same user, have shown anti-war messages aimed towards Russia, such as the one given below.



```
This code serves as a non-destructive example of why controlling
your node modules is important. It also serves as a non-violent
protest against Russia's aggression that threatens the world right
now. This module will add a message of peace on your users'
desktops, and it will only do it if it does not already exist
just to be polite.
```

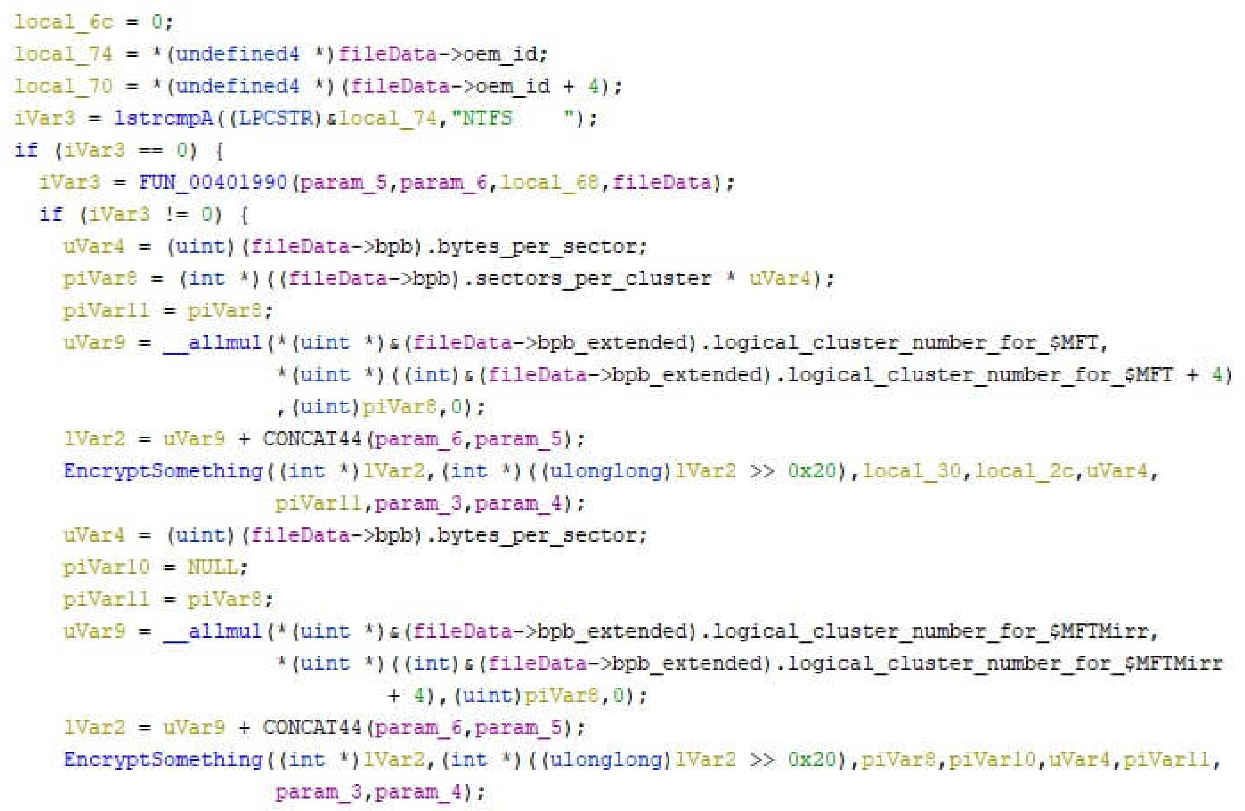
Figure 2 - The activist's message, image courtesy of Snyk

Aside from activism, Advanced Persistent Threats (APTs) can utilize wipers to wreak havoc. Many of the wipers which targeted Ukrainian entities have been attributed to pro-Russian actors by a variety of security companies. The return-on-investment for a wiper can be high, given the low effort that is required to create one. However, not all wipers are created equal, as some are more complex than others. HermeticWiper shows how its creator(s) got an understanding of NTFS and FAT file systems, iterating and altering the file system in a more advanced manner than simply deleting or overwriting all encountered files. The images below show a segment of the NTFS parsing code within the wiper, before and after assigning the correct data types during the analysis. These structures aren't included in most analysis programs by default and take additional time to be found and created. The additionally required time benefits the attackers, as their goal is to slow the defenders down as much as possible. The two images below show the "fileData" variable, first with raw offsets, and secondly with the applied structure names.



```
local_6c = 0;
local_74 = *(undefined4 *)((int)fileData + 3);
local_70 = *(undefined4 *)((int)fileData + 7);
iVar2 = lstrcmpA((LPCSTR)&local_74,"NTFS    ");
if (iVar2 == 0) {
  iVar2 = FUN_00401990(param_5,param_6,local_68,fileData);
  if (iVar2 != 0) {
    uVar3 = (uint)*(ushort *)((int)fileData + 0xb);
    piVar7 = (int *)(*(byte *)((int)fileData + 0xd) * uVar3);
    piVar10 = piVar7;
    uVar8 = __allmul(fileData[0xc],fileData[0xd],(uint)piVar7,0);
    lVar1 = uVar8 + CONCAT44(param_6,param_5);
    EncryptSomething((int *)lVar1,(int *)((ulonglong)lVar1 >> 0x20),local_30,local_2c,uVar3,
                     piVar10,param_3,param_4);
    uVar3 = (uint)*(ushort *)((int)fileData + 0xb);
    piVar9 = NULL;
    piVar10 = piVar7;
    uVar8 = __allmul(fileData[0xe],fileData[0xf],(uint)piVar7,0);
    lVar1 = uVar8 + CONCAT44(param_6,param_5);
    EncryptSomething((int *)lVar1,(int *)((ulonglong)lVar1 >> 0x20),piVar7,piVar9,uVar3,piVar10,
                     param_3,param_4);
```

Figure 3 - Pseudo code of NTFS related objects in HermeticWiper without the correct types



```
local_6c = 0;
local_74 = *(undefined4 *)fileData->oem_id;
local_70 = *(undefined4 *)(fileData->oem_id + 4);
iVar3 = lstrcmpA((LPCSTR)&local_74,"NTFS    ");
if (iVar3 == 0) {
  iVar3 = FUN_00401990(param_5,param_6,local_68,fileData);
  if (iVar3 != 0) {
    uVar4 = (uint)(fileData->bpb).bytes_per_sector;
    piVar8 = (int *)((fileData->bpb).sectors_per_cluster * uVar4);
    piVar11 = piVar8;
    uVar9 = __allmul(*(uint *)&(fileData->bpb_extended).logical_cluster_number_for_$MFT,
                     *(uint *)((int)&(fileData->bpb_extended).logical_cluster_number_for_$MFT + 4)
                     ,(uint)piVar8,0);
    lVar2 = uVar9 + CONCAT44(param_6,param_5);
    EncryptSomething((int *)lVar2,(int *)((ulonglong)lVar2 >> 0x20),local_30,local_2c,uVar4,
                     piVar11,param_3,param_4);
    uVar4 = (uint)(fileData->bpb).bytes_per_sector;
    piVar10 = NULL;
    piVar11 = piVar8;
    uVar9 = __allmul(*(uint *)&(fileData->bpb_extended).logical_cluster_number_for_$MFTMirr,
                     *(uint *)((int)&(fileData->bpb_extended).logical_cluster_number_for_$MFTMirr
                              + 4),(uint)piVar8,0);
    lVar2 = uVar9 + CONCAT44(param_6,param_5);
    EncryptSomething((int *)lVar2,(int *)((ulonglong)lVar2 >> 0x20),piVar8,piVar10,uVar4,piVar11,
                     param_3,param_4);
```

Figure 4 - Pseudo code of NTFS related objects in HermeticWiper with the correct types

The destruction of the file system, rather than files within the file system, makes it harder to simply restore some files to “repair” the victimized machine. A complete back-up needs to be restored instead, which takes more time, especially if this needs to be done in bulk.

Target selection

The selection of the target(s) from an actor’s point of view mainly depends on two factors. First, the nature of the attack. Where APT groups are often inclined to remain undetected, activists usually want to make their cause known, relying on the media to amplify their message.

Even though the reasoning is different, there is a parallel to be drawn with regards to other malware attacks where targeted attacks are carried out by more sophisticated groups. Massively spread malware is generally classified as commodity malware, and while both can have a disastrous impact, their modus operandi of spreading varies.

The second factor is the targeted operating system. Windows is the most used platform for corporate networks, whereas various Linux distributions are often used to host servers. Wiping files from employee machines already impacts the functioning of a business and can be done with little effort as it requires no escalation of privileges.

Most of the wipers that were observed in this research target the Windows operating system. However, using a different platform as a defense against wipers is not a solution, as some of the observed wipers target a very specific niche. Examples of such niches are the wiping of Viasat KA-SAT modems with the AcidRain wiper, as reported by SentinelOne. Other examples are the Shred wipers, as observed by ESET, which target Linux distributions as well as the Solaris operating system.

Delivery methods

One way or another, actors want to execute the malware of their choice on the victim's machine(s). There are numerous ways to do so, and the below is not meant as an exhaustive list, but rather the most seen methods that were used to deliver the wipers.

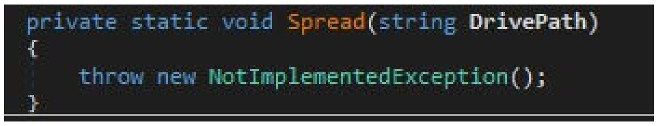
One execution strategy that has been observed is manual execution of the wipers, either per device or via the group policy to execute it on numerous machines at once. The HermeticWiper has been executed via a scheduled task on the victimized devices, as reported by Symantec.

Additionally, or alternatively, actors can create a worm-like spreading mechanism to execute the wiper on all devices it can connect to. HermeticWizard has such a capability, which deploys HermeticWiper, as explained by ESET.

Similarities and useful differences

While some wipers contain overlap in the code, it's not always a link to a specific actor. Two examples where it is likely that the same actor is behind the code are the RURansom and its alleged development version dubbed dnWipe, and the two different versions of the CaddyWiper. The former two were discovered by TrendMicro, whereas the latter was found by ESET.

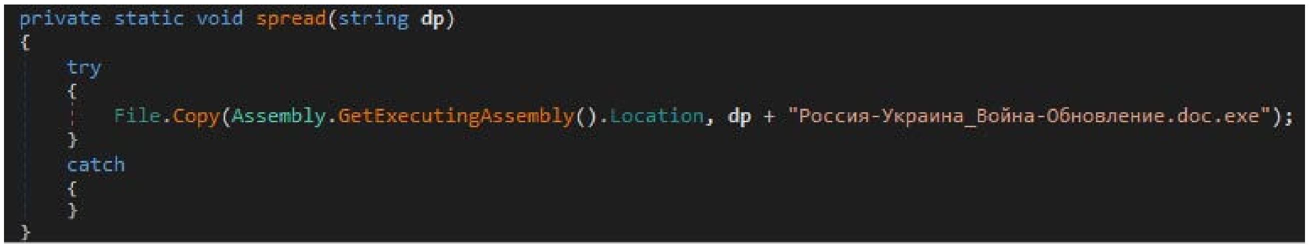
The dnWipe sample, although it only base64 encodes files rather than wiping, has several to-be-implemented functions, one of which can be seen below.



```
private static void Spread(string DrivePath)
{
    throw new NotImplementedException();
}
```

Figure 5 - dnWipe's lacking spread function

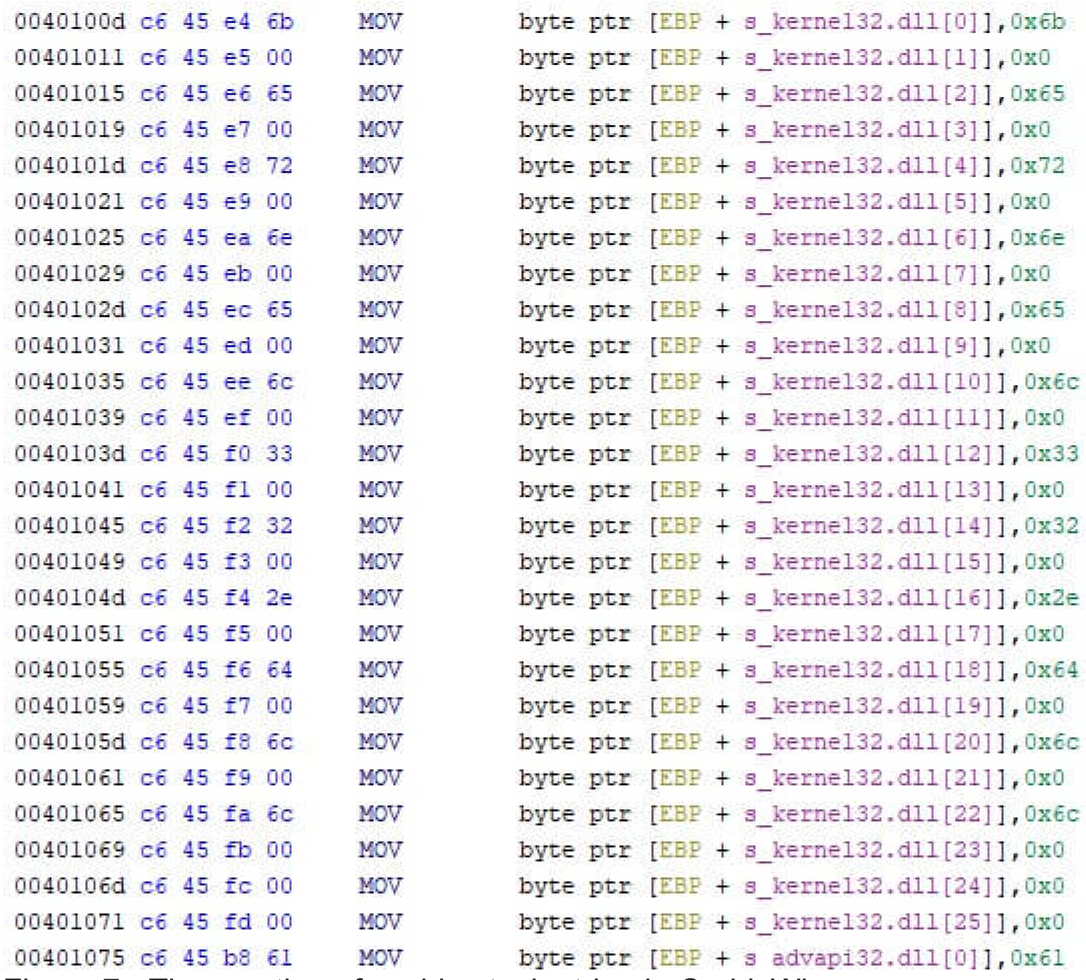
The RURansom wiper uses very similar code, which is unlikely to be copied from dnWipe by another actor due to its minimal spreading. The image below shows the implementation of the spread function.



```
private static void spread(string dp)
{
    try
    {
        File.Copy(Assembly.GetExecutingAssembly().Location, dp + "Россия-Украина_Война-Обновление.doc.exe");
    }
    catch
    {
    }
}
```

Figure 6 - RURansom's implemented spread function

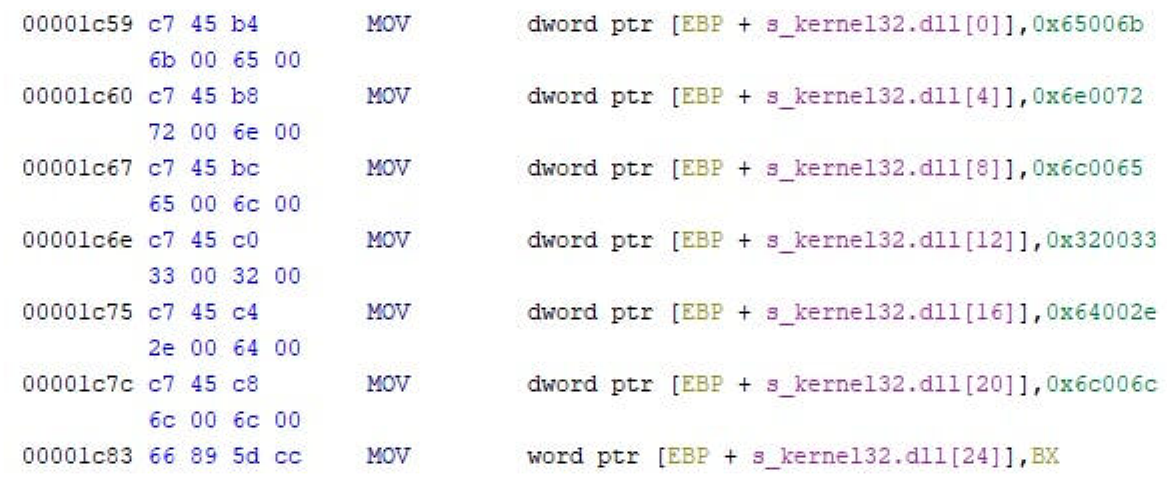
CaddyWiper was used twice. Where the two versions contain minor, yet useful, differences related to the used assembly instructions, the malware's behavior hasn't changed. The initial version built the (wide) stack-strings character by character, as seen in the screenshot below.



```
0040100d c6 45 e4 6b     MOV        byte ptr [EBP + s_kernel32.dll[0]],0x6b
00401011 c6 45 e5 00     MOV        byte ptr [EBP + s_kernel32.dll[1]],0x0
00401015 c6 45 e6 65     MOV        byte ptr [EBP + s_kernel32.dll[2]],0x65
00401019 c6 45 e7 00     MOV        byte ptr [EBP + s_kernel32.dll[3]],0x0
0040101d c6 45 e8 72     MOV        byte ptr [EBP + s_kernel32.dll[4]],0x72
00401021 c6 45 e9 00     MOV        byte ptr [EBP + s_kernel32.dll[5]],0x0
00401025 c6 45 ea 6e     MOV        byte ptr [EBP + s_kernel32.dll[6]],0x6e
00401029 c6 45 eb 00     MOV        byte ptr [EBP + s_kernel32.dll[7]],0x0
0040102d c6 45 ec 65     MOV        byte ptr [EBP + s_kernel32.dll[8]],0x65
00401031 c6 45 ed 00     MOV        byte ptr [EBP + s_kernel32.dll[9]],0x0
00401035 c6 45 ee 6c     MOV        byte ptr [EBP + s_kernel32.dll[10]],0x6c
00401039 c6 45 ef 00     MOV        byte ptr [EBP + s_kernel32.dll[11]],0x0
0040103d c6 45 f0 33     MOV        byte ptr [EBP + s_kernel32.dll[12]],0x33
00401041 c6 45 f1 00     MOV        byte ptr [EBP + s_kernel32.dll[13]],0x0
00401045 c6 45 f2 32     MOV        byte ptr [EBP + s_kernel32.dll[14]],0x32
00401049 c6 45 f3 00     MOV        byte ptr [EBP + s_kernel32.dll[15]],0x0
0040104d c6 45 f4 2e     MOV        byte ptr [EBP + s_kernel32.dll[16]],0x2e
00401051 c6 45 f5 00     MOV        byte ptr [EBP + s_kernel32.dll[17]],0x0
00401055 c6 45 f6 64     MOV        byte ptr [EBP + s_kernel32.dll[18]],0x64
00401059 c6 45 f7 00     MOV        byte ptr [EBP + s_kernel32.dll[19]],0x0
0040105d c6 45 f8 6c     MOV        byte ptr [EBP + s_kernel32.dll[20]],0x6c
00401061 c6 45 f9 00     MOV        byte ptr [EBP + s_kernel32.dll[21]],0x0
00401065 c6 45 fa 6c     MOV        byte ptr [EBP + s_kernel32.dll[22]],0x6c
00401069 c6 45 fb 00     MOV        byte ptr [EBP + s_kernel32.dll[23]],0x0
0040106d c6 45 fc 00     MOV        byte ptr [EBP + s_kernel32.dll[24]],0x0
00401071 c6 45 fd 00     MOV        byte ptr [EBP + s_kernel32.dll[25]],0x0
00401075 c6 45 b8 61     MOV        byte ptr [EBP + s advapi32.dll[0]],0x61
```

Figure 7 - The creation of a wide stack-string in CaddyWiper

The second version, also originally observed by ESET and obtained after dumping it from memory, built the (wide) stack-strings using double words whenever possible, as shown in the screenshot below. Note that both wide stack-strings are taken from the entry points of both samples.



```
00001c59 c7 45 b4       MOV       dword ptr [EBP + s_kernel32.dll[0]],0x65006b
         6b 00 65 00
00001c60 c7 45 b8       MOV       dword ptr [EBP + s_kernel32.dll[4]],0x6e0072
         72 00 6e 00
00001c67 c7 45 bc       MOV       dword ptr [EBP + s_kernel32.dll[8]],0x6c0065
         65 00 6c 00
00001c6e c7 45 c0       MOV       dword ptr [EBP + s_kernel32.dll[12]],0x320033
         33 00 32 00
00001c75 c7 45 c4       MOV       dword ptr [EBP + s_kernel32.dll[16]],0x64002e
         2e 00 64 00
00001c7c c7 45 c8       MOV       dword ptr [EBP + s_kernel32.dll[20]],0x6c006c
         6c 00 6c 00
00001c83 66 89 5d cc    MOV       word ptr [EBP + s_kernel32.dll[24]],BX
```

Figure 8 - The creation of a wide stack-string in CaddyWiper II

Such minor (and likely compiler related) changes often indicate source code level access by whoever compiled the malware. It is, however, easy to mistakenly attribute source code level access to the same actor. The code could have leaked, or could be widely available within a nation state’s arsenal, as is the case with PlugX.

Note that minor changes in the code might not always be related to source code access, as files can be patched to alter their behavior, as is the case with WannaCry’s many edited variants still roaming around with edited kill switches, ready to permanently encrypt devices.

Attribution, and difficulties thereof

Based on the attribution issues that are described above, there is an observation to make: the actor who uses the malware can be very different form the actor which creates the malware. This is already seen when looking at ransomware affiliates, who use the ransomware, but rarely write it themselves. With regards to nation state actors, it could very well be possible for actors to contract software engineers to create a program for them. This would make code-based attribution even harder than it already is, as different contractors have vastly different coding styles, ranging from subtle nuances all the way to the usage of different programming languages altogether.

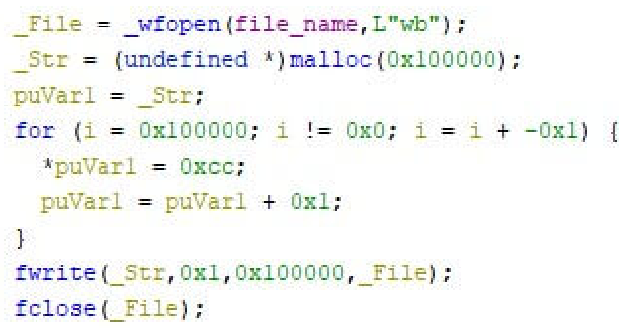
While there is no bullet proof evidence for this hypothesis, it does provide food for thought. Not all wipers look the same, though multiple were used in Ukraine, likely stemming from a single nation state with regards to geopolitical issues, even though code-based evidence is lacking. As mentioned in June 2022, we observed an actor who intended to wipe the systems of a victim with the WhisperGate wiper, but failed in doing so. A second attempt used the HermeticWiper instead. Although this is only anecdotal evidence given the public availability of the wipers at the time, it is the exact scenario of the above-described hypothesis.

Recovery options

Even though the term “wiper” might lead one to assume that files are deleted from the targeted system, this isn’t necessarily the case. The wiper’s goal is to cripple the system, which can also be done by overwriting files. Note that disk type specifics are omitted here for the sake of brevity, as well as different file systems. Most the wipers focus on Windows, which has been using NTFS as the default file system for well over a decade.

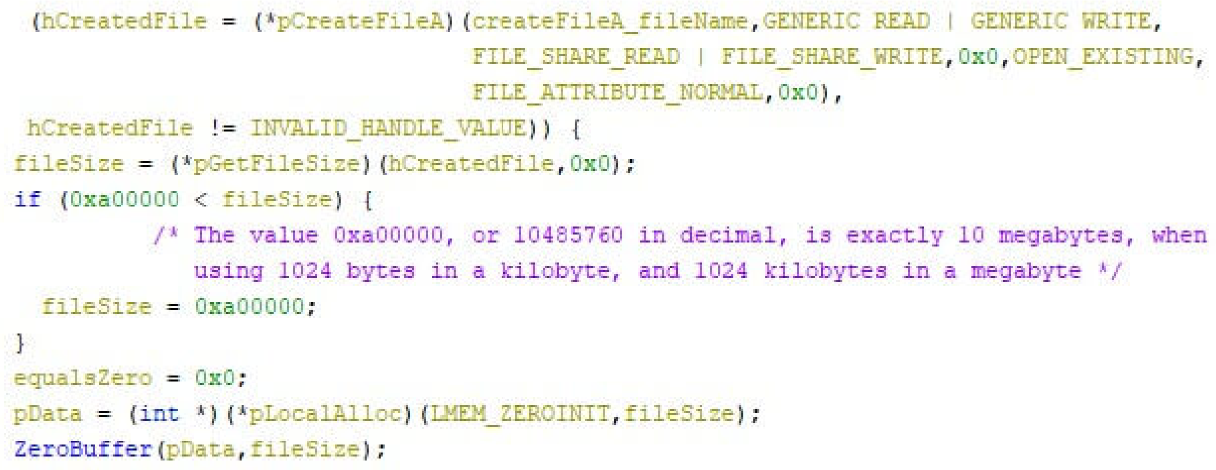
There are different reasons as to why one would choose to overwrite files over the deletion of files. Whoever wrote the wiper might have simply chosen one or the other without giving it much thought, but there is a difference when attempting to recover files. Deleting a file often leaves the file on the disk as-is while marking the size as free-to-use for new write operations. As such, the deletion of all files from the device leaves a lot of free disk space, while preserving a lot of data. When carving the disk, numerous files can be recovered this way.

The WhisperGate wiper, however, shows few advanced mechanisms, as the targeted files were corrupted by overwriting the first megabyte of each file with 0xCC repeatedly. Similarly, both CaddyWiper variants use zero files, whereas RURansom uses a random encryption key per file, and Nominatus_ToxicBattery (as reported by G-Data) wipes the system by overwriting files with copies of itself. The images below show excerpts from the mentioned samples. As some files are larger than the targeted file sizes, it is possible that files are corrupted, rather than overwritten. In either case, the files remain unusable.



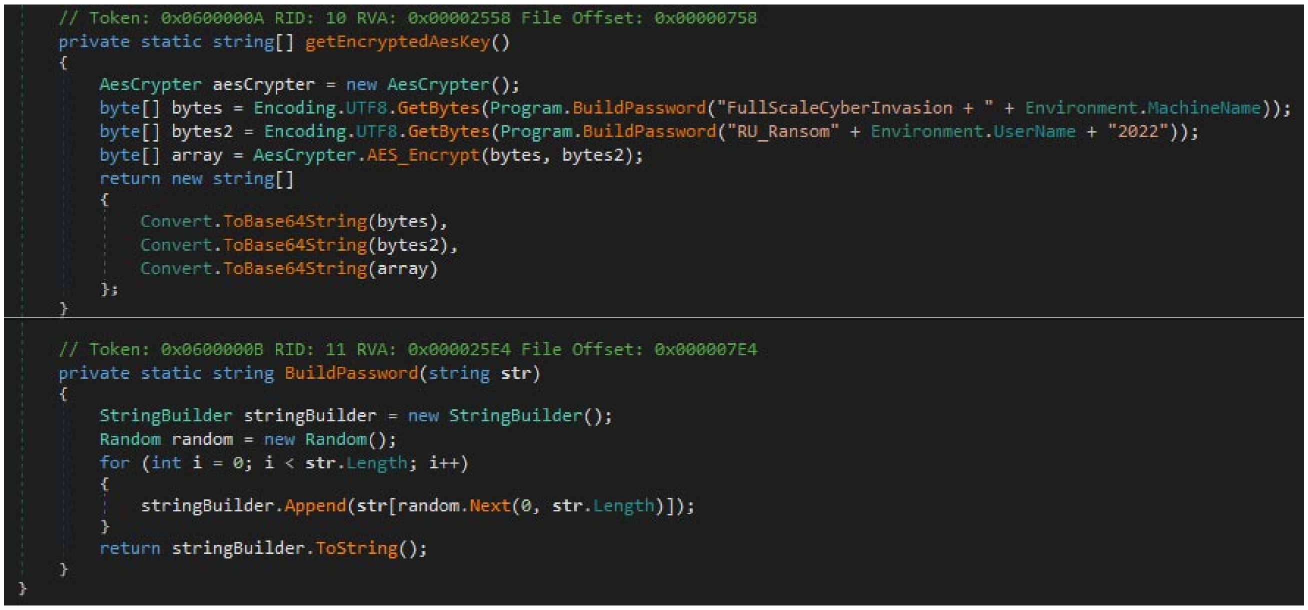
```
_File = _wfopen(file_name,L"wb");
_Str = (undefined *)malloc(0x100000);
puVar1 = _Str;
for (i = 0x100000; i != 0x0; i = i + -0x1) {
  *puVar1 = 0xcc;
  puVar1 = puVar1 + 0x1;
}
fwrite(_Str,0x1,0x100000,_File);
fclose(_File);
```

Figure 9 - WhisperGate's wiper's pseudocode to overwrite the first megabyte with 0xCC



```
 (hCreatedFile = (*pCreateFileA)(createFileA_fileName,GENERIC_READ | GENERIC_WRITE,
                                 FILE_SHARE_READ | FILE_SHARE_WRITE,0x0,OPEN_EXISTING,
                                 FILE_ATTRIBUTE_NORMAL,0x0),
 hCreatedFile != INVALID_HANDLE_VALUE)) {
fileSize = (*pGetFileSize)(hCreatedFile,0x0);
if (0xa00000 < fileSize) {
          /* The value 0xa00000, or 10485760 in decimal, is exactly 10 megabytes, when
             using 1024 bytes in a kilobyte, and 1024 kilobytes in a megabyte */
  fileSize = 0xa00000;
}
equalsZero = 0x0;
pData = (int *)(*pLocalAlloc)(LMEM_ZEROINIT,fileSize);
ZeroBuffer(pData,fileSize);
```

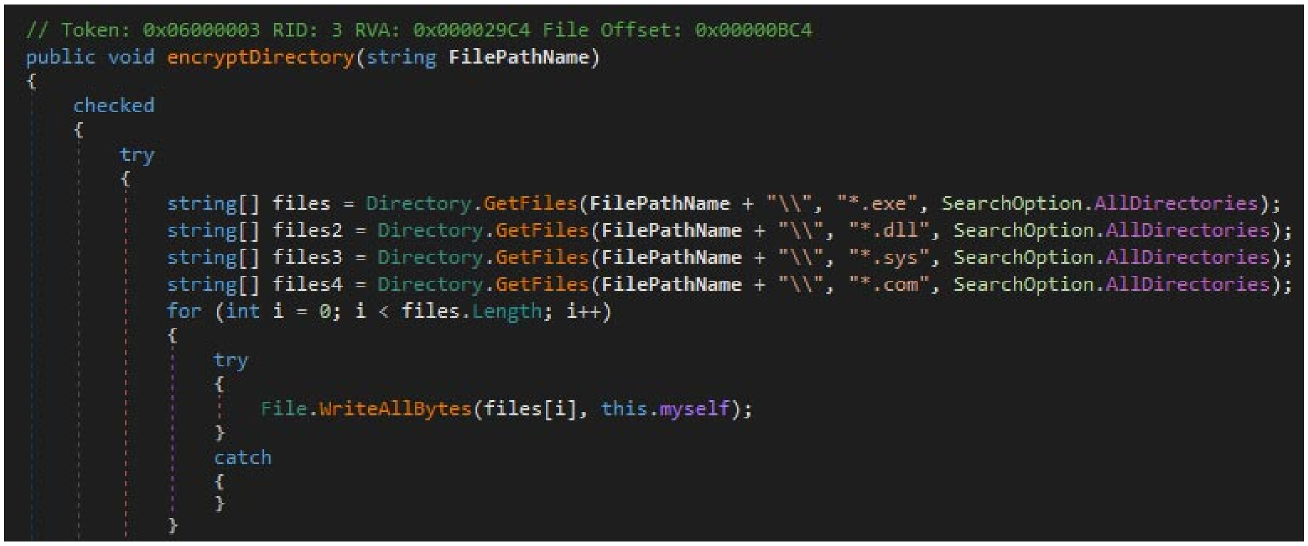
Figure 10 - CaddyWiper (both I and II) overwriting the first 10 megabytes of targeted files with zeros



```
    // Token: 0x0600000A RID: 10 RVA: 0x00002558 File Offset: 0x00000758
    private static string[] getEncryptedAesKey()
    {
        AesCrypter aesCrypter = new AesCrypter();
        byte[] bytes = Encoding.UTF8.GetBytes(Program.BuildPassword("FullScaleCyberInvasion + " + Environment.MachineName));
        byte[] bytes2 = Encoding.UTF8.GetBytes(Program.BuildPassword("RU_Ransom" + Environment.UserName + "2022"));
        byte[] array = AesCrypter.AES_Encrypt(bytes, bytes2);
        return new string[]
        {
            Convert.ToBase64String(bytes),
            Convert.ToBase64String(bytes2),
            Convert.ToBase64String(array)
        };
    }

    // Token: 0x0600000B RID: 11 RVA: 0x000025E4 File Offset: 0x000007E4
    private static string BuildPassword(string str)
    {
        StringBuilder stringBuilder = new StringBuilder();
        Random random = new Random();
        for (int i = 0; i < str.Length; i++)
        {
            stringBuilder.Append(str[random.Next(0, str.Length)]);
        }
        return stringBuilder.ToString();
    }
}
```

Figure 11 - RURansom's random password generation, called per file



```
// Token: 0x06000003 RID: 3 RVA: 0x000029C4 File Offset: 0x00000BC4
public void encryptDirectory(string FilePathName)
{
    checked
    {
        try
        {
            string[] files = Directory.GetFiles(FilePathName + "\\", "*.exe", SearchOption.AllDirectories);
            string[] files2 = Directory.GetFiles(FilePathName + "\\", "*.dll", SearchOption.AllDirectories);
            string[] files3 = Directory.GetFiles(FilePathName + "\\", "*.sys", SearchOption.AllDirectories);
            string[] files4 = Directory.GetFiles(FilePathName + "\\", "*.com", SearchOption.AllDirectories);
            for (int i = 0; i < files.Length; i++)
            {
                try
                {
                    File.WriteAllBytes(files[i], this.myself);
                }
                catch
                {
                }
            }
```

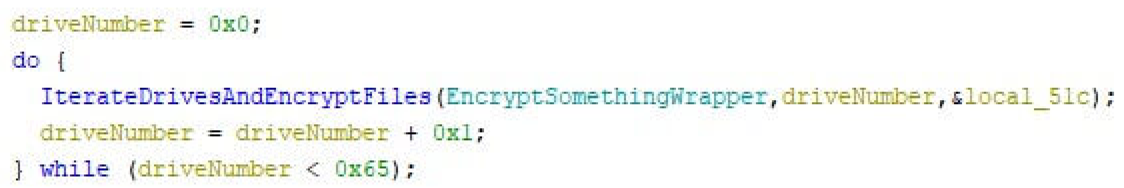
Figure 12 - Nominatus_ToxicBattery's "encryption" function where it replaces files with a copy of itself

This is not to say that overwritten files cannot be recovered. The reorganizing of files on the disk, along with overwriting files, can overwrite the free-to-use disk segments, although this is no certainty. Even with (partially) overwritten disk segments, file recovery tools can sometimes manage to recover most, if not all, of the lost data.

The recovery of data ties in with the next step, which is related to back-ups.

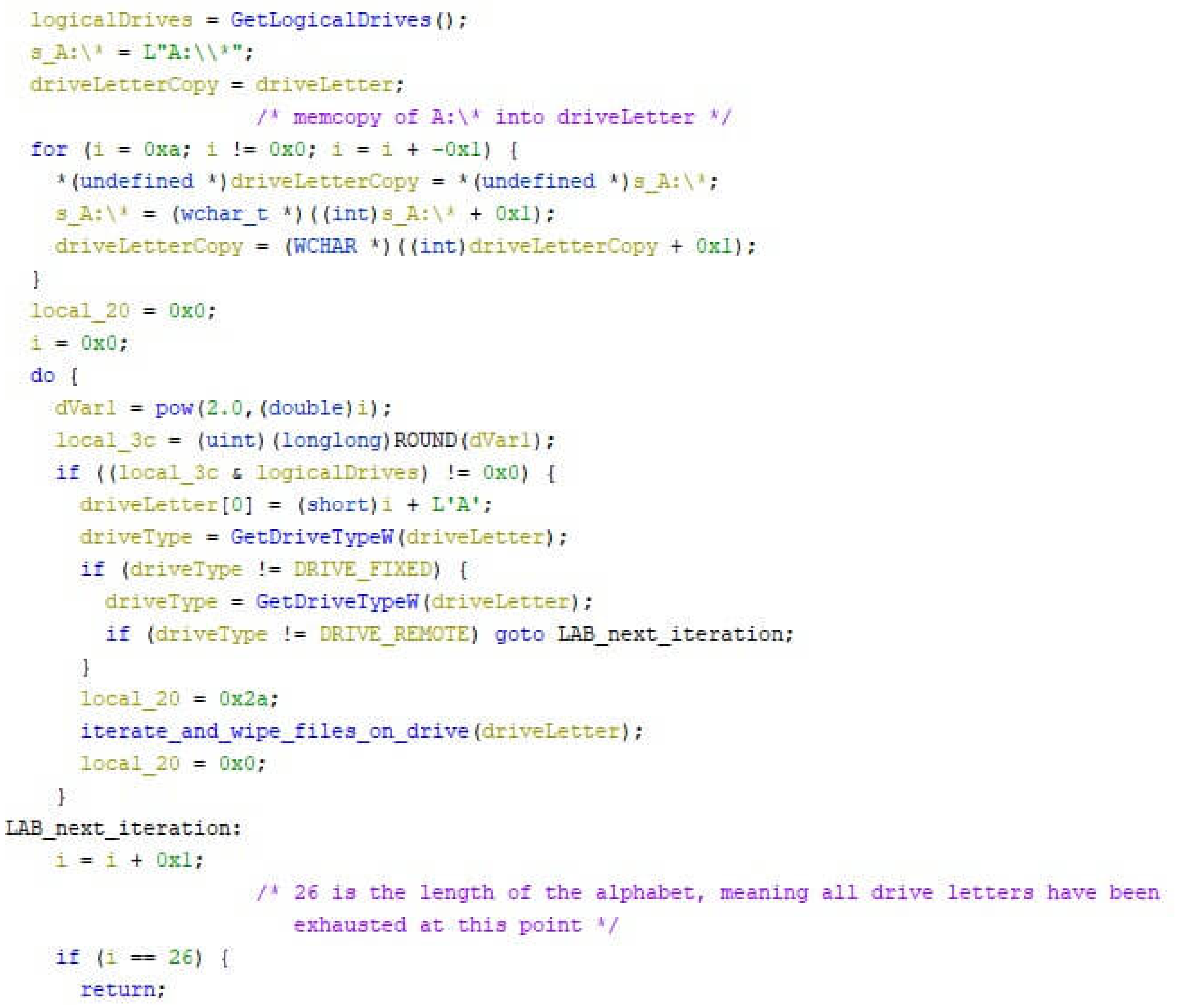
Restoring back-ups

The destruction of wipers is often not limited to the device’s main drive, which is usually C: on Windows. The 26 letter bound drives can be targeted as well, as is the case with both the WhisperGate wiper and the HermeticWiper. This can significantly increase the amount of data which has to be restored from the backup, increasing the wiper’s impact.



```
driveNumber = 0x0;
do {
  IterateDrivesAndEncryptFiles(EncryptSomethingWrapper,driveNumber,&local_51c);
  driveNumber = driveNumber + 0x1;
} while (driveNumber < 0x65);
```

Figure 13 - HermeticWiper iterating over drive numbers



```
 logicalDrives = GetLogicalDrives();
 s_A:\* = L"A:\\*";
 driveLetterCopy = driveLetter;
                  /* memcopy of A:\* into driveLetter */
 for (i = 0xa; i != 0x0; i = i + -0x1) {
   *(undefined *)driveLetterCopy = *(undefined *)s_A:\*;
   s_A:\* = (wchar_t *)((int)s_A:\* + 0x1);
   driveLetterCopy = (WCHAR *)((int)driveLetterCopy + 0x1);
 }
 local_20 = 0x0;
 i = 0x0;
 do {
   dVar1 = pow(2.0,(double)i);
   local_3c = (uint)(longlong)ROUND(dVar1);
   if ((local_3c & logicalDrives) != 0x0) {
     driveLetter[0] = (short)i + L'A';
     driveType = GetDriveTypeW(driveLetter);
     if (driveType != DRIVE_FIXED) {
       driveType = GetDriveTypeW(driveLetter);
       if (driveType != DRIVE_REMOTE) goto LAB_next_iteration;
     }
     local_20 = 0x2a;
     iterate_and_wipe_files_on_drive(driveLetter);
     local_20 = 0x0;
   }
LAB_next_iteration:
   i = i + 0x1;
                  /* 26 is the length of the alphabet, meaning all drive letters have been
                     exhausted at this point */
   if (i == 26) {
     return;
```

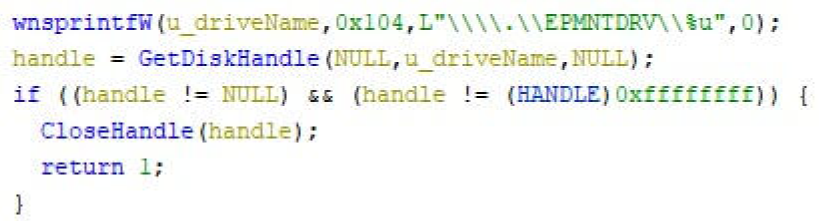
Figure 14 - WhisperGate's wiper iterating logical drives to wipe

The restoration speed can be too slow, especially when people work from home and use their private broadband access to download significant amounts of data. Additionally, the uplink of the back-up location is limited until a certain point. If the restoration covers hundreds of machines, the uplink cannot handle all machines concurrently at a decent speed, or it might not even support the concurrent number at all.

The slow restoration speed for a few machines might not seem more than a hinderance, but if the number of affected machines rises, so does the restoration time, creating a longer than anticipated downtime once the initial malware infection has been remediated.

Benign drivers

The industry’s constant dilemma is to create detection rules without false positives. The Shamoon wiper used a benign driver to manipulate the file system. Since the driver was signed and used in a manner which is expected for such a driver, it is difficult to heuristically define the software as malicious. When looking at the type of file system changes, it becomes obvious that the program is malware. On the 23rd of February 2022, HermeticWiper was first discovered. This wiper also used a benign driver, one with a similar purpose as the one Shamoon used, as is shown in the image below.



```
wnsprintfW(u_driveName,0x104,L"\\\\.\\EPMNTDRV\\%u",0);
handle = GetDiskHandle(NULL,u_driveName,NULL);
if ((handle != NULL) && (handle != (HANDLE)0xffffffff)) {
  CloseHandle(handle);
  return 1;
}
```

Figure 15 - The usage of a benign driver in HermeticWiper

Safety measures

The measures below are not meant as an exhaustive list, but rather as a few key points blue teams can use to further strengthen their infrastructure against wipers, as well as other similarly operating malware such as ransomware.

Much like ransomware, wipers interact with thousands of files in a matter of minutes. As such, a detection rule to flag processes which remove, rename, or rewrite hundreds of files within a short timespan can help. Note that depending on the ingested logs, such a rule might be prone to false positives, as creating compressed archives (be it encrypted or not) with a lot of files will match the same characteristics.

Additionally, or alternatively, some wipers might simply delete all the files they encounter, including shadow copies and event logs. These two are normally not deleted, nor completely overwritten, making them interesting objects to monitor. The same advice applies to the Master Boot Record (MBR), which is only changed rarely, if at all.

The following advice is not only useful when one or more machines fall victim to a wiper: create and test back-ups. The back-up system should only be connected to the machines when storing the back-up, or it is at risk of being affected by various types of malware, not only wipers. Ransomware tends to encrypt files in all connected drives, thus including attached back-up drives.

The last advice in this section, is to restrict the privilege of users to the minimum requirements to do their job. Administrative privileges can change the impact of the wiper from losing files to not being able to boot the machine. In either case, it is likely for the machine to be reimaged, but it does make the restoration process lengthier as the laptop cannot be restored remotely, and the victim must find another way of contacting the Security Operations Center.

Conclusion

All in all, wipers have proven to be effective. A wiper's heuristic behavior can look quite a lot like ransomware, with an equal, if not higher, initial devastating impact. For individuals who are stuck, the impact likely depends on the quality of the available back-ups, if any. For corporations and nation states alike, the impact depends on the number of affected machines, and the moment in time when the incident occurs. Communication is vital during wartime, which is why the impact of the wipers in Ukraine could have been disastrous, even if back-ups could be restored within a few days without any hiccups, as the few days with limited communication might give an opponent the edge.

Wipers can be effective regardless of the technical skills of the attacker, as even the simplest wiper can wreak havoc on affected systems. The required time to create such a piece of malware is low, especially when compared to complex espionage backdoors and the often-accompanying vulnerabilities that are used. The return of investment need not be high in those cases, although it is unlikely that a few wipers are to wreak that much havoc in and of themselves.

The security posture of an organization can only benefit from preparing against such a disastrous event, as less impactful yet similar events can be mitigated using the same playbook. As the saying goes: forewarned is forearmed.

Hashes

In this section, hashes of the wipers which were mentioned in detail are shared below, in their order of appearance. Note that the linked articles refer to the same samples, but they've been aggregated here for transparency.

HermeticWiper

SHA-256	1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591
SHA-1	61b25d11392172e587d8da3045812a66c3385451
MD-5	3f4a16b29f2f0532b7ce3e7656799125

dnWipe

SHA-256	610ec163e7b34abd5587616db8dac7e34b1aef68d0260510854d6b3912fb0008

SHA-1	fbeb9eb14a68943551b0bf95f20de207d2c761f6
MD-5	191e51cd0ca14edb8f06c32dcba242f0

RURansom

SHA-256	107da216ad99b7c0171745fe7f826e51b27b1812d435b55c3ddb801e23137d8f
SHA-1	a30bf5d046b6255fa2c4b029abbcf734824a7f15
MD-5	8fe6f25fc7e8c0caab2fdca8b9a3be89

CaddyWiper I

SHA-256	a294620543334a721a2ae8eaaf9680a0786f4b9a216d75b55cfd28f39e9430ea
SHA-1	98b3fb74b3e8b3f9b05a82473551c5a77b576d54
MD-5	42e52b8daf63e6e26c3aa91e7e971492

CaddyWiper II

SHA-256	7f76e7a9e784b90463a67ad40b1acf68c6e706fe489f82058ae608dbc203f832
SHA-1	b903014ade8b2c19e18cfd366b0dd8670e8747a6
MD-5	87f906fe3d7be0f0f941a59ffa41dd27

WhisperGate wiper

SHA-256	191ca4833351e2e82cb080a42c4848cfbc4b1f3e97250f2700eff4e97cf72019
SHA-1	8be3c66aecd425f1f123aadc95830de49d1851b5
MD-5	343fcded2aaf874342c557d3d5e5870d

Nominatus_ToxicBattery

SHA-256	45e433d6fd0710d2905f21fda25c02fccab9eef43732384f0f0ea65ee464b936
SHA-1	d91ae3b5ac290e0687d02605a4a3168823da5943
MD-5	2dd15c0d305242a89a35c8b61e4398ff